Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A process for executing programs on at least one processor having a given instruction set architecture, the process comprising the operations of:

compiling the program to be executed and translating said program into native instructions of said instruction set architecture, organizing the instructions deriving from the translation of said program into respective bundles arranged in an order of successive bundles, each bundle grouping together instructions adapted to be executed in parallel by said at least one processor;

separating said bundles of instructions into respective sub-bundles by detecting a value of a binary symbol encoded in one of the instructions of the respective bundle, said sub-bundles identifying a first set of instructions, which must be executed before the instructions belonging to the next bundle of said order, and a second set of instructions that can be executed both before and in parallel with respect to the instructions belonging to said next bundle of said order, it being possible for at least said second set of instructions to be the null set;

defining a sequence of execution of the instructions of said sub-bundles in successive operating cycles of said at least one processor, while preventing, in assigning each sub-bundle to an operating cycle of the processor, simultaneous assignment, to the same operating cycle, of two sub-bundles corresponding to instructions belonging to said first set of two successive bundles of said order; and

executing said instructions on at least one said processor respecting said execution sequence.

2. (Previously Presented) The process according to Claim 1, further comprising the operation of selectively varying the overall length of instruction executed for each cycle by said at least one processor.

3. (Canceled)

4. (Currently Amended) The process according to Claim ~~3,~~ 1, further comprising the operations of:

detecting when said second set is the null set; and

inserting in the respective sub-bundle a fictitious instruction which does not imply any execution of operations.

5. (Currently Amended) The process according to Claim 1, wherein the binary symbol identifies the last instruction of the first set of a respective bundle and the separating said bundles of instructions into respective sub-bundles further comprises ~~comprising the operation of identifying the instructions belonging to a sub-bundle of said first set and of said second set by means of two~~ detecting a value of a second ~~distinct~~ binary ~~symbols~~ symbol encoded in a second instruction of the respective bundle, wherein the second binary symbol ~~which identify~~ identifies the last instruction of the respective sub-bundle.

6. (Previously Presented) The process according to Claim 1, for executing programs on a multiprocessor system comprising a plurality of processors having said instruction-set architecture, further comprising the operations of:

instantiating the processors of said plurality with respective degrees of parallelism of execution with at least two different values of said parallelism of execution in the context of said plurality; and

selectively distributing execution of the instructions of said sequence of execution among the processors of said plurality, the instructions of said sequence of execution being directly executable by the processors of said plurality in conditions of binary compatibility.

7. (Previously Presented) The process according to Claim 6, further comprising the operation of selectively distributing the execution of the instructions of said sequence among the processors of said plurality, dynamically distributing the computational load of said processors.

8. (Previously Presented) The process according to Claim 6, further comprising the operation of selectively distributing the execution of the instructions of said sequence among said processors of said plurality with the criterion of equalizing the operating frequency of the processors of said plurality.

9. (Previously Presented) The process according to Claim 6, further comprising the operation of performing a process of control executed by at least one of the processors of said plurality so as to equalize its own workload with respect to the other processors of said multiprocessor system.

10. (Previously Presented) The process according to Claim 9, further comprising the operation of drawing up a table accessible by said control process, said table having items chosen from the group made up of:

a list of processes that are being executed or are suspended on any processor of said plurality of processors;

the progressive number thereof according to the order of activation;

the percentage of maximum power of the processor that is used by said process;

the execution time;

the amount of memory of the system used by said process to be able to execute the function for which it is responsible;

the processor on which the process currently resides; and

the address of the portion of memory in which the data and the instructions are stored.

11. (Original) A processor system, preferably of a multiprocessor type, configured for operating with the process according to Claim 1.

12. (Currently Amended) A process of executing programs on a system having a plurality of processors comprising:

compiling the program to be executed;

translating said program into instruction sets;

organizing said instruction sets into respective groups, each group having a predetermined priority for execution in a given processor of said plurality;

separating each group of instructions into a respective first sub-bundle of instructions which must be executed before the instructions belonging to the next group, and a respective second sub-bundle of instructions that can be executed before or in parallel with respect to the instructions belonging to said next group, it being possible for at least said second sub-bundle of instructions to be a null set;

encoding said instructions for execution on said processors; and

providing in each encoded instruction a designated number of initial bits identifying said predetermined priority of the instruction set.

13. (Original) The process of Claim 12, wherein the execution of programs comprises directing of the instruction sets to said processors of said plurality according to the priority bits encoded into the said instruction set.

14. (Original) The process of Claim 12, wherein said priority is determined based on the amount of memory required by each of the processors of said plurality to execute said instruction set.

15. (Original) The process of Claim 12, wherein said priority is determined based on the amount of percentage of maximum power required by each of the processors of said plurality to execute said instruction set.

16. (Currently Amended) A system comprising:

a plurality of processors coupled for receiving instruction sets, each instruction set containing one or more instructions; and

a first processor of the plurality coupled to an instruction ~~stream,~~ stream and capable of directing said instruction sets to each of the processors of said plurality for execution;

said first processor configured to direct ~~directing~~ the instructions sets to ~~each of~~ the processors of said plurality based on ~~the~~ priority values carried by a designated number of bits encoded into each instruction.

17. (Original) The system according to Claim 16, wherein the priority for the instruction sets is based on the amount of memory required by each of the processors of said plurality to execute said instruction set.

18. (Original) The system according to Claim 16, wherein the priority for the instruction sets is based on the amount of percentage of maximum power required by each of the processors of said plurality to execute said instruction set.

19. (Canceled)